Research Review: Mastering the game of Go with deep neural networks and tree search

Médéric Hurier

Summary

In this paper, the authors present the architecture of AlphaGo, an artificial agent capable of beating skilled human players and other programs at the game of Go. Historically, the game of Go has been one of the most challenging classical game for artificial agents. Indeed, the high branching factor and search depth of the game have limited prior works to shallow evaluations of game states. Due to these constraints, classical algorithms such as depth-first minimax search and alpha-beta pruning are not able to perform exhaustive searches, even at small search depth. Thus, artificial agents have not been able to surpass the best human players at the game. Compared these works, AlphaGo is able to perform well, even in the most difficult settings against professional players, with an overall reported winning rate of 99.8%.

The architecture of the program is composed of several neural networks with an architecture similar to deep convolutional networks. Their goal is to evaluate the situation of the current player (value networks) and select its next action accordingly (policy networks). Mainly two techniques are used to create these networks: a Supervised Learning based approach (SL) that trains the network from recorded game sessions, and a Reinforcement Learning (RL) based approach that refines the agent training from self-played games. On one hand, SL based networks provide a quick feedback and high quality gradients to bootstrap the agent. On the other hand, RL based networks are able to refine the training of the system and optimize its game play. AlphaGo then combines the policy and value networks using the Monte Carlo Tree Search (MCTS). This technique has been used in prior works to perform lookahead tree search. The search tree is traversed by simulation in order to find the action that maximizes the expected value for the player. Because of its high computation requirements, the final version of the AlphaGo has to leverage both CPU, GPU and possibly distributed computing to perform these simulations.

To evaluate AlphaGo, the authors organized several tournaments both against internal variants of AlphaGo and external programs (CrazyStone, Zen, Pachi, Fuego and GnuGo). Out of 495 game played, AlphaGo was able to win 494 games. Even in handicap games where opponents are allowed four free moves, the winning rate of AlphaGo exceeds 77%. The distributed version of AlphaGo was also able to win against Fan Hui, a professional 2 dan player, 5 games against 0.

Review

In conclusion, the model and performance of AlphaGo are impressive. The main contribution of the architecture is through the combination of different machine learning techniques along with efficient hardware usage. The authors were also able to use less handcrafted knowledge than other state-of-the-art Go programs in their solution. Similar to what DeepBlue was to artificial intelligence, AlphaGo also should mark an important progress to the field. The contribution of the authors is significant and should lead to further investigation in other domains beyond classical game play. I particularly enjoyed the capacity of the authors to combine existing techniques based on their strengths and weaknesses.